Copy No. _____

Defence Research and Development Canada
Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

# AIMSsim version 2.3.4

## *System Manual*

*Oliver Schoenborn, Patrick Lachance and Nima Bahramifarid*
*CAE Professional Services*

*CAE Professional Services*
*1135 Innovation Drive, Suite 300*
*Ottawa, Ontario K2K 3G7*

*Project Manager: O. Schoenborn, 613-247-0342*
*Contract Number: W7711-047904/TOR/001*
*Contract Scientific Authority: J. Crebolder, 902-426-3100 x296*

**Defence R&D Canada – Atlantic**

Contract Report
DRDC Atlantic CR 2007-301
January 2008

Canada

This page intentionally left blank.

# AIMSsim version 2.3.4

*System Manual*

Oliver Schoenborn, Patrick Lachance and Nima Bahramifarid
CAE Professional Services

CAE Professional Services
1135 Innovation Dr., Suite 300
Ottawa, ON
K2K 3G7

Project Manager: Oliver Schoenborn, 613-247-0342

Contract number: W7711-047904/TOR/001

Contract Scientific Authority: J. Crebolder, 902-426-3100 x296

## Defence R&D Canada – Atlantic

## Contract Report

DRDC Atlantic CR 2007-301

January 2008

Author

Oliver Schoenborn

Approved by

*Original signed by Jacquelyn Crebolder*

Jacquelyn Crebolder

Approved for release by

*Original signed by James L. Kennedy*

James L. Kennedy

# Abstract

This system manual provides an overview of software developed to support the empirical investigation of a simulated user interface for an Advanced Integrated Multi-sensor Surveillance (**AIMS**) system (formerly known as the Enhanced Low-Light Level Visible and Infrared Surveillance System – **ELVISS**). The AIMS system is an electro-optical imaging system being developed by Defence Research and Development Canada (**DRDC**) – Valcartier to enhance the capability of search and rescue (**SAR**) crews to operate effectively at night and in degraded weather conditions. In order to ensure that a SAR operator would be able to use the system effectively and with a minimal amount of training, a prototype human-machine interface (**HMI**) was developed to permit evaluating design concepts. The latest development phase added sensor controller options and a configurable display interface for evaluating interface design concepts.



# Résumé

Le présent manuel de système donne un aperçu du logiciel élaboré pour prendre en charge l'examen empirique d'une interface utilisateur simulée d'un système perfectionné de surveillance multi-capteurs intégré (**AIMS**) (anciennement appelé Système perfectionné de surveillance à intensification de lumière visible et à infrarouge ou **ELVISS**). Le système AIMS est un système d'imagerie électro-optique en cours de développement à Recherche et développement pour la défense Canada (**RDDC**) – Valcartier pour améliorer la capacité des équipages de recherche et de sauvetage (**SAR**) à travailler avec efficacité la nuit et par intempérie. Afin de garantir qu'un opérateur du service SAR sera en mesure d'utiliser le système efficacement et moyennant une formation minimale, un prototype d'interface humain-machine (**IHM**) a été développé pour évaluer les principes de conception. La dernière phase de développement consistait à ajouter des options de contrôleur des capteurs et une interface d'affichage configurable pour évaluer les principes de conception d'interface.

This page intentionally left blank

# Executive summary

## Introduction

This document provides an overview of software developed to support the empirical investigation of a simulated user interface for an Advanced Integrated Multi-sensor Surveillance (**AIMS**) system (formerly known as the Enhanced Low-Light Level Visible and Infrared Surveillance System – **ELVISS**).

Defence Research and Development Canada (**DRDC**) is developing a multi-sensor system composed of an Active Gated TV (**AGTV**) and a thermal Infrared (**IR**) imaging system. By coordinating the use of a pulsed laser illuminator and AGTV camera, the AGTV component of AIMS provides effective imaging in the absence of ambient light. In addition, the active range gate allows the AGTV system to penetrate meteorological phenomena such as fog, snow and rain much more effectively than a FLIR camera. The FLIR camera is a passive thermal imaging system that produces an image based on temperature variation by detecting mid-infrared and far-infrared radiation. Both sensors are bore sighted and are packaged in a gimballed "ball" that is mounted on the exterior of an aircraft or vehicle. The use of gyros inside the ball allow the camera within to maintain its orientation in the Earth frame of reference, without being affected by roll, pitch and yaw changes in the supporting aircraft or vehicle.

In order to ensure that a SAR operator would be able to use the sensor suite effectively and with a minimal amount of training, a prototype human-machine interface (HMI) was developed to provide a cost effective method for evaluating design concepts and assessing the impact of design characteristics on operator performance. The prototype was developed for the Silicon Graphics, Inc.'s (**SGI**) platform but was eventually ported to run on Microsoft Windows. A series of modifications to the software have enhanced the capability of the research platform. This report document additions in the latest development phases, which include tracking and motion-related functionality, graphical user interface (GUI) display configurability, a larger number of sensors/displays, a thumbnail bar, and a new name, AIMSsim.

## Results

In the latest rendition the system, now called AIMSsim, has been further modified to include:

- A capability to over-ride the default joystick controller on the fly panel with a joystick connected via a USB port to allow an evaluation of different types of controllers (e.g., displacement joystick; force feedback joystick)
- A feature that allows the display interface to be completely reconfigured, to accommodate test and evaluation of interface design concepts since the sensor suite now supports five sensors instead of two as previously contained in ELVISS.
- The ability to change the position of the sensor image windows on the interface, relocate the moving map display, and add thumbnails representing each of the sensors. A variety of display options can be modelled.

## Significance

The continued development and upgrade of the *AIMSsim* research platform provides the experimenter with an appropriate level of simulation detail to conduct human performance analyses which in turn delivers up-to-date knowledge and advice on the design of sensor surveillance systems to the military stakeholder. A User Manual (Schoenborn, 2007a) describing functionality and how to use the system, and a Final Report (Schoenborn, 2007b) that summarizes the work performed are also associated with this document.

Schoenborn O., 2007; AIMSsim Feature Development II. DRDC Atlantic CR 2007-301. Defence R&D Canada – Atlantic.

# Sommaire

## Introduction

Le présent document donne un aperçu du logiciel élaboré pour prendre en charge l'examen empirique d'une interface utilisateur simulée d'un système perfectionné de surveillance multi-capteurs intégré (**AIMS**) (anciennement appelé Système perfectionné de surveillance à intensification de lumière visible et à infrarouge ou **ELVISS**).

Recherche et développement pour la défense Canada (**RDDC**) est en cours de développer un système multi-capteurs composé d'un capteur de télévision commandée par portes actives (**AGTV**) et d'un système d'imagerie thermique infrarouge (**IR**). En coordonnant l'utilisation d'un illuminateur laser et d'une caméra AGTV, l'élément AGTV de l'AIMS assure une imagerie efficace en l'absence de lumière ambiante. De plus, la porte active de distance permet au système AGTV de pénétrer des phénomènes météorologiques tels que le brouillard, la neige et la pluie beaucoup plus efficacement qu'une caméra FLIR. La caméra FLIR est un système d'imagerie thermique passive qui produit une image en fonction de la variation de la température en détectant le rayonnement infrarouge moyen et le rayonnement infrarouge lointain. Les deux capteurs sont à simbleau et sont montés sur cardan dans une « boule » installée à l'extérieur d'un aéronef ou d'un véhicule. L'utilisation de gyros à l'intérieur de la boule permet à la caméra de maintenir son orientation par rapport à la Terre, sans être influencé par les mouvements de roulis, de tangage et de lacet de la plate-forme d'installation.

Afin de garantir qu'un opérateur SAR sera capable d'utiliser l'ensemble de capteurs efficacement et moyennant une formation minimale, un prototype d'interface humain-machine (IHM) a été développé pour fournir une méthode économique d'évaluer les principes de conception et d'évaluer l'incidence des caractéristiques de conception sur le rendement de l'opérateur. Le prototype avait été développé pour la plate-forme de Silicon Graphics Inc. (**SGI**), mais on a fini par le faire migrer vers Microsoft Windows. Une série de modifications au logiciel a amélioré les capacités de la plate-forme de recherche. Le présent rapport décrit les ajouts qui ont été apportés pendant les dernières phases de développement et qui comprennent les fonctions en rapport avec la poursuite et le mouvement, la configurabilité de l'affichage de l'interface graphique (GUI), un plus grand nombre de capteurs/afficheurs, une barre de vignettes et un nouveau nom : AIMSsim.

## Résultats

La dernière version du système, appelé maintenant AIMSsim, comprend les fonctions suivantes :

- Capacité de surpasser le contrôleur à manette par défaut sur le boîtier de commande FlyPanel au moyen d'une manette connectée à un port USB pour permettre l'évaluation de différents types de contrôleurs (p. ex. manette de commande de déplacement; manette à retour de force)

- Fonction qui permet la reconfiguration complète de l'interface d'affichage, pour prendre en charge l'essai et l'évaluation des principes de conception, puisque l'ensemble de capteurs compte maintenant cinq capteurs au contraire de celui du système ELVISS, qui n'en comptait que deux.
- Capacité de changer la position des fenêtres d'image de capteur sur l'interface, de déplacer l'affichage cartographique dynamique et d'ajouter des vignettes représentant chacun des capteurs. Possibilité de modéliser une variété d'options d'affichage.

## Portée

Le développement et la mise à niveau continus de la plate-forme de recherche *AIMSsim* offrent à l'expérimentateur un niveau de simulation assez détaillé pour mener des analyses du rendement humain, qui fournissent à leur tour à l'intervenant militaire des connaissances à jour et des conseils au sujet de la conception de systèmes de surveillance à capteurs. Un manuel de l'utilisateur (Schoenborn, 2007a), décrivant les fonctions et le mode d'emploi du système, et un rapport final (Schoenborn, 2007b), qui résume le travail effectué, sont également associés au présent document.

Schoenborn O., 2007; AIMSsim Feature Development II. DRDC Atlantic CR 2007-301. Defence R&D Canada – Atlantic.

## Contributors

| | | |
|---|---|---|
| Various | 1999 to Feb 2005 | CMC Electronics, The HFE Group, DRDC Toronto |
| Oliver Schoenborn | Feb – Mar 2005 | Greenley & Associates |
| Oliver Schoenborn, Bassem Mikhael | Apr – Oct 2005 | Greenley & Associates |
| Oliver Schoenborn | Dec 2005 – Aug 2007 | CAE Professional Services |

**Foreword:**

This version of the System Manual incorporates all changes to the DRDC AIMS HMI Experimental Prototype, as of June 2007. This document is based on the System Manual originally created by the HFE Group, and the Experimenter's Guide, provided by DRDC. The HFE Group and CMC, early contributors to this document, are no longer responsible for its content. Along with the User's Manual, this manual provides a good introduction and reference to the DRDC AIMS HMI Experimental Prototype software, AIMSsim.

# Table of contents

## List of Figures

# List of Tables

This page intentionally left blank

# Introduction

## 1.1 General

Defence Research and Development (DRDC) Valcartier has been developing a flyable prototype of an Advanced Integrated Multi-sensor Surveillance (**AIMS**) system, formerly known as the Enhanced Low-Light-Level Visible and Infrared Surveillance System – **ELVISS**.

In conjunction with the flyable prototype DRDC has been contracting out the development and enhancement of the capabilities for a research platform simulating the human-machine interface (**HMI**) of ELVISS and subsequently AIMS. The research platform allows for the empirical investigation of interface and sensor characteristics on search and detection capability under different simulated environmental conditions.

(For details of the ELVISS interface and operator control panel currently used in the flyable prototype refer to McFadden (2006), and to Baker & Youngston (2006) for a report on interface design concepts for AIMS.)

## 1.2 Background

The Department of National Defence (**DND**) has identified a requirement to enhance the capabilities of Search and Rescue (**SAR**) operators to conduct operations at night and under degraded weather conditions. To this end, Defence Research and Development Canada (**DRDC**) is developing a multi-sensor system composed of an Active Gated TV (**AGTV**) and a thermal Infrared (**IR**) imaging system. By coordinating the use of a pulsed laser illuminator and AGTV camera, the AGTV component of AIMS provides effective imaging in the absence of ambient light. In addition, the active range gate allows the AGTV system to penetrate meteorological phenomena such as fog, snow and rain much more effectively than a FLIR camera. The FLIR camera is a passive thermal imaging system that produces an image based on temperature variation by detecting mid-infrared and far-infrared radiation. Both sensors are bore sighted and are packaged in a gimballed "ball" that is mounted on the exterior of an aircraft or vehicle. The use of gyros inside the ball allows the camera within to maintain its orientation in the Earth frame of reference, without being affected by roll, pitch and yaw changes in the supporting aircraft or vehicle.

In order to ensure that a SAR operator would be able to use the system effectively and with a minimal amount of training, a prototype HMI was developed to evaluate design concepts. The VAPS HMI prototype (**ELVISS**), developed for the Silicon Graphics, Inc.'s (**SGI**) platform, provided a cost effective method for evaluating the impact of design characteristics of dual sensor systems on operator performance. However the capability was limited and the architecture was not designed to support systematic investigation of the usefulness of the proposed system under different conditions or to manipulate the sensor and interface characteristics.

The ELVISS VAPS prototype was therefore extensively enhanced to allow the empirical investigation of different interface and sensor characteristics on search and detection capability under different environmental conditions. Included in this upgrade was a Scenario Generation Environment (SGE) that provided user-friendly capability for generating scenarios. Nonetheless, despite the increased versatility of the prototype the requirements of a specific experimental design required that LUA scripting be used to make additional modifications to the software. While further development drastically expanded the LUA scripting capabilities, the SGE was not similarly extended and some of its scenario-generation capabilities became incompatible with the prototype. Thus LUA scripting in now the primary mode of control of the prototype.

The prototype HMI was then ported to run on Microsoft Windows. This involved replacing the VAPS and SGI Performer™ with equivalent functionality using the OpenSceneGraph open source graphics library. The robustness and traceability of the system were significantly improved. The latest development phases added (amongst other things) important tracking and motion-related functionality, GUI display configurability, a larger number of EO and IR sensors/displays, a thumbnail bar. Since 2006, the software system is known as AIMSsim.

## 1.3 Goal

The aim of this manual is to provide an overview of the system software, AIMSsim, developed to support empirical investigation of a simulated user interface for the AIMS system.

## 1.4 Objectives

The specific objectives of this manual are to:

a. Describe the capabilities and limitations of the software developed to support the conduct of experimentation using the AIMS software prototype;

b. Define the minimum requirements of the hardware and software environment needed to utilize the software developed for this project; and

c. Provide information about various subsystems and algorithms used.

## 1.5 Report Outline

The report is structured as follows:

a. Section One describes the background, aim and scope of this document;

b. Section Two describes the architecture of the developed software;

c. Section Three and Four explain capabilities and the limitations, respectively, of the developed software;

d. Section Five defines the hardware and software requirements for the execution and/or modification of the developed software;

e. Section Six documents the subsystems and some algorithms of the developed software; and

f. The annexes document abbreviations used in this manual and a support tool called the “Scenario Generation Environment”.

# 2 Architecture

The AIMSsim is divided into three separate applications, as sketched in Figure 1. The *simControl* application, as the name would suggest, controls the execution of the experiment and simulation. It provides a state machine and a LUA scripting interface to describe experiments, and provides data collection facilities. Also, it "models" the world physics relevant to the simulation, such as vehicle (aircraft and targets) motion, and is the first responder to operator controls.

The *simDisplay* application is an implementation of the AIMS User Interface (UI), built using OpenSceneGraph (which itself sits atop OpenGL). It handles all visual aspects of the HMI: rendering the sensor displays and the moving map display (**MMD**), providing dialog screens, etc. In the displays, it is in charge of computing and overlaying such information as the current zoom factor, the simulation time, the height above terrain, the map scale, the sensor footprint or search history, etc.

The third application is *simInputs*, which is in charge of reading the hardware via a serial port connection, packages the data and makes it available to *simControl* via shared memory.

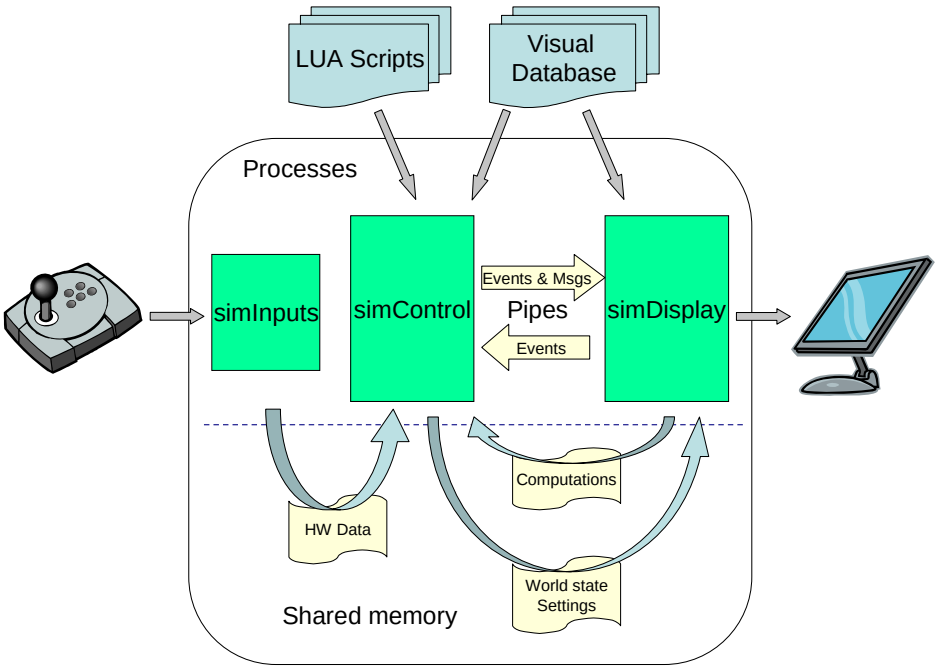


***Figure 1: AIMSsim architecture***

The behavior of the UI is controlled remotely by *simControl*, and as such *simDisplay* can be thought of as a display-only application. It has no knowledge of how the virtual world behaves, only how it looks visually. However it is more than a "dumb display"

since it is able to compute quantities related to geometry, such as line-of-sight intersection with the terrain.

Both *simInputs* and *simDisplay* are automatically started when *simControl* is run. Hardware state data from *simInputs* is read by *simControl* via shared memory. The state of the world and what type of information or screen to show in *simDisplay* is also communicated from *simControl* to *simDisplay* via shared memory. In some instances, *simDisplay* makes some of its computations available to *simControl* via a read-only segment of shared memory, for use by the experiment scripts for data acquisition or decision making. Note however that *simControl* uses its own copy of the terrain geometry for time-critical computations. Finally, *simControl* and *simDisplay* send and receive events from each other via POSIX pipes (one in each direction). Events include those resulting from operator input via mouse or hardware (pressing a trigger etc), those generated by the experiment scripts, and system events indicating something has happened.

The *simControl* is configured by an initialization script that defines the runtime settings and the various states available to the experiment, via a programmable Finite State Machine (**FSM**), described further below. This initialization script may also define targets and path plans, and assign path plans to the aircraft and targets. Similarly, *simDisplay* can be configured via an initialization script that can be used to create new dialog screens and configure the Operational View display. The simDisplay script does not involve an FSM.

Both programs use *Lua* as their scripting language. Lua is a relatively simple language that is interpreted at run time and uses a C-like syntax. While it is easy to use, it is at the same time a very powerful language. For information about Lua look at the appendix in the existing AIMSsim User's Guide, or visit www.lua.org for official documentation.

The FSM itself defines which scripts to run when certain events occur, and what is the resulting state name (or stage) of the experiment. Part of a script is shown in Figure 2 as an example.

```
Set("baseTerrain", "Nerepis/shaded/ELVISS_ne_smaller.ive")

PathPlanCreate("aircraftPath")
PathPlanAddWaypoint("aircraftPath", 7946, 8434, 584, 100)
PathPlanAddWaypoint("aircraftPath", 6446, 8434, 584, 150, 50)
PathPlanAddWaypoint("aircraftPath", 6446, 9934, 584, 100)
PathPlanAddWaypoint("aircraftPath", 7946, 9934, 584, 150, 400)
PathPlanAddWaypoint("aircraftPath", 7946, 8434, 584, 150)

AircraftAddPath("aircraftPath")
ReadTargetsFile("TEST/sge_targets.xml")

TargetChangeAttrib("target 1", "retroReflective", true)
TargetChangeAttrib("target 3", "colorOverrideFLIR", "yes")
TargetChangeAttrib("target 3", "colorInFLIR", 0)
```

***Figure 2. Example of part of a LUA initialization script***

The visual database used by simDisplay includes terrain geometry, targets geometry, and OpenGL *shader* programs. The latter are text files that are used to model various aspects of sensor displays. For instance, the effects of fog, laser illuminator, noise, retro-reflectivity, and thermal imaging are modelled using such shaders. It is relatively straightforward to change those shader files using any standard text editor, to obtain new visual effects, without having to rebuild the software.

# 3 Capabilities

## 3.1 General

The HMI prototype, AIMSsim, allows the dynamic configuration and conduct of an experimental Search and Rescue scenario via LUA scripts and a Finite State Machine (FSM), and provides data collection capabilities. An AIMSsim Scenario Generation Environment (**SGE**) is available to ease the generation of path plans and the positioning of targets.

## 3.2 AIMS Human Machine Interface (HMI) Prototype

AIMSsim, the AIMS HMI Prototype simulates the AIMS user interface. The HMI Prototype utilizes text scripts to construct and execute a simulated scenario. A simulated scenario can involve

- the visualization of a scenario landscape (or terrain) as viewed through several Electro-Optic (**EO**) and Infra-Red (**IR**) sensor displays
- the visualization of a moving map display (**MMD**) with SAR-related symbology such as sensor footprint, search history, etc
- the population of the terrain with "target" objects with various characteristics, at any time in the experiment
- the definition of path plans that can be strung together, to define the motion of vehicles (targets and aircraft)
- the flight of a simulated search aircraft over the terrain, using any sequence of the defined path plans
- the configuration and layout of the UI itself, e.g. where the MMD and sensors get displayed, which dialog screens will be available, etc
- the management, control and monitoring of operator input
- the acquisition, processing and saving of experiment data

### 3.2.1 MMD and sensor displays

The MMD is able to show such information as the sensor footprint, search history, map scale in use, flight path, North indicator, aircraft location and direction, above-ground level, and designation markers, whereas the sensor displays simulate the effects of noise, fog, laser illuminator, (and interactions of those phenomena) and are

overlaid with useful information such as the current zoom factor, depth of line of sight to terrain, etc.

### 3.2.2 Scripting and FSM

The HMI Prototype incorporates a powerful scripting engine providing a programmable Finite State Machine (FSM), advanced data collection capabilities, dynamic scenarios that can change as the experiment progresses, and the potential for external control via Semi-Automated Forces (SAF) components.

The data collection capability allows the experimenter to capture important information about the system state, the state of simulated entities, and the interaction between the operator and the prototype, throughout the execution of an experimental scenario. It allows to further process the data in any way that can be described by the scripting language, and to save it in any format encodable by the experimenter.

The FSM allows for a structured decomposition of the experiment into a series of repeatable and reusable steps, or stages, and provides a powerful way of allowing only certain operator inputs or events. Functionality from one experiment can be re-used by other experiments, with a minimal amount of work.

A FSM can be viewed as a graph with the nodes representing states, and the arcs representing state transitions, as shown in Figure 3. By describing all the transitions, the stages of an experiment can be fully described. There are only two states required by the AIMSsim system, “INIT” and “EXIT”, i.e. the start and end points of the experiment. All states in between can be named as desired by the experimenter.

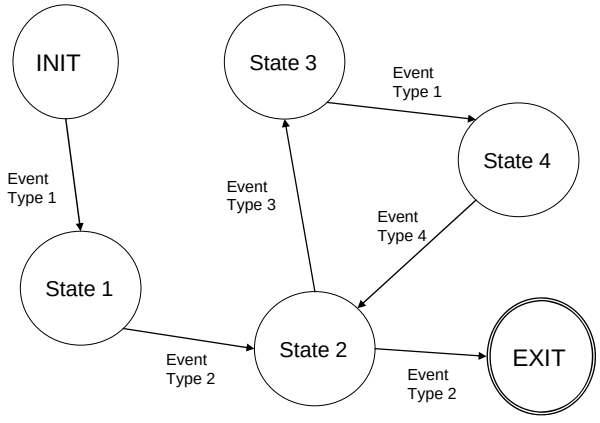


***Figure 3: Example FSM of a scenario***

The FSM that describes an experiment must be set at startup time through the Lua initialization script given to *simControl*. This allows for any ordering of screens and

behaviour to be created. The AIMSsim system should be able to behave according to any desired experimental flow using the scripting facilities provided. It should be noted, however, that care must be taken in describing the FSM of an experiment, since there is no facility to ensure that the FSM described makes sense, or that all properties of a state are set to values appropriate to the state

### 3.2.3 Operator input

Operator input is based on the FlyPanel by BG Systems Inc. This input device is continuously scanned by simInputs and its state made available to simControl. The FlyPanel is shown in Figure 4 and Figure 5.

***Figure 4: FlyPanel by BG Systems Inc.***

***Figure 5: Digital inputs on FlyPanel joystick***

The operator has access to a 2D joystick to pan and tilt the virtual camera pod, and to two levers to control the zoom factor on the primary and auxiliary sensor displays. All other inputs are digital and must be defined through the FSM: each input has an associated event name which can be used to trigger a transition to a different state or stage of the experiment.

The joystick component of the FlyPanel can be overridden by an external USB joystick connected to the PC. This should allow for a larger variety of joystick devices to be tested for usability. Only the stick position, as well as the controls on the FlyPanel joystick, are overridden; all other FlyPanel controls state read is unaffected.

### 3.2.4 Scripting in simDisplay (GUI)

The display side of AIMSsim, provided by simDisplay, can be configured via a *displayConfig.lua* script present in experiment folder. This script is read only at startup and provides the following main capabilities:

1. New "dialog" screens can be created, consisting of text and buttons and events. They can be activated (i.e. made visible to the operator) via the simControl experiment scripts, using `SendMessage("toDlgNAME")` where `NAME` is the name given to the dialog when created.

2. The Operational View's default "output ports" can be overridden to specify where/if each component should be displayed. E.g. the default “ops displays” is shown in Figure 6, whereas one of many dozen possible modifications is shown in Figure 7.

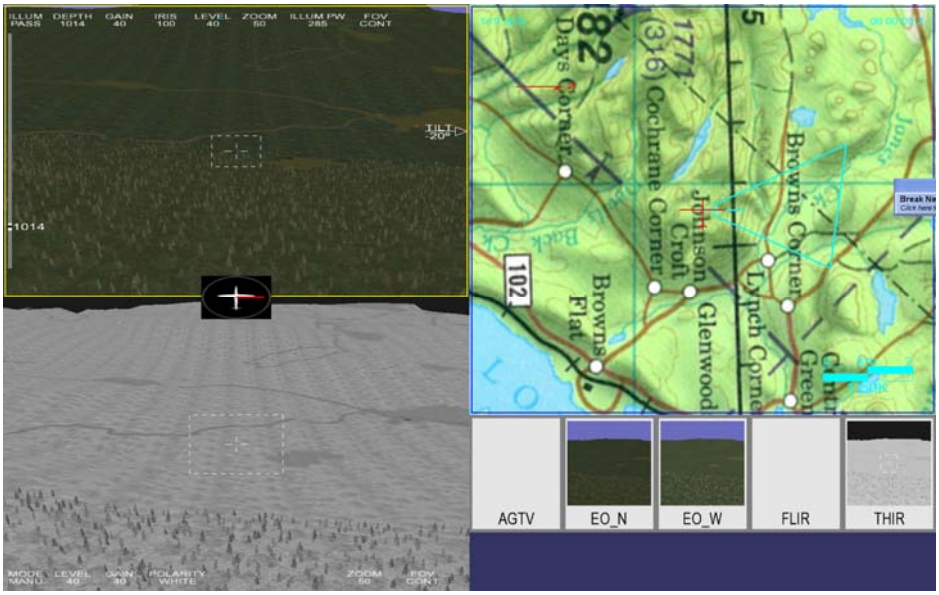


***Figure 6: Default Operational View configuration***


***Figure 7: Example of Operational View configuration from displayConfig.lua***

### 3.2.5 ATD Cues

Automatic Target Detection is partially supported in the system, in the form of visual cues in the AGTV sensor display, shown at the location of visible targets, as shown in Figure 8. The User Manual discusses how to activate this capability via the AIMS_ATD_CUES environment variable and how to setup ATD cues in an experiment. Currently there are 3 types of cues available: Box (red), Image overlay, and Inverted Color.

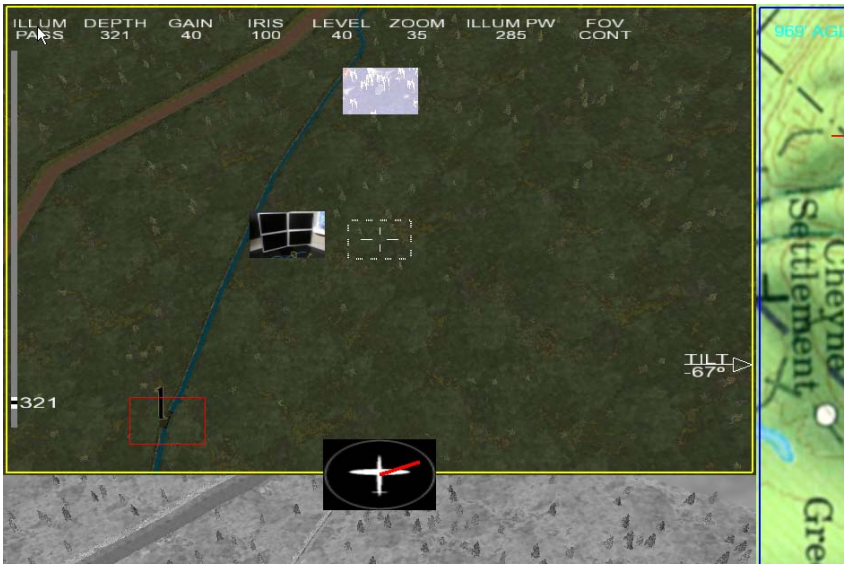


***Figure 8: The three types of ATD cues available.***

## 3.3 Scenario Generation Environment (SGE)

The SGE is a GUI application that can be used as a visual aid to define targets and place them in the world terrain, and to define flight plans, as it provides 1D, 2D, and 3D views of the world and targets. Two of the four files generated by the SGE can be read by AIMSsim's *simControl*, namely the targets definition file and the flight plan file (Figure 9). The SGE also enables the creation of several predefined types of flight plans. The flight plans created can be loaded into path plans by the AIMSsim scripts, and used on *any* vehicle (aircraft and targets), at any time.

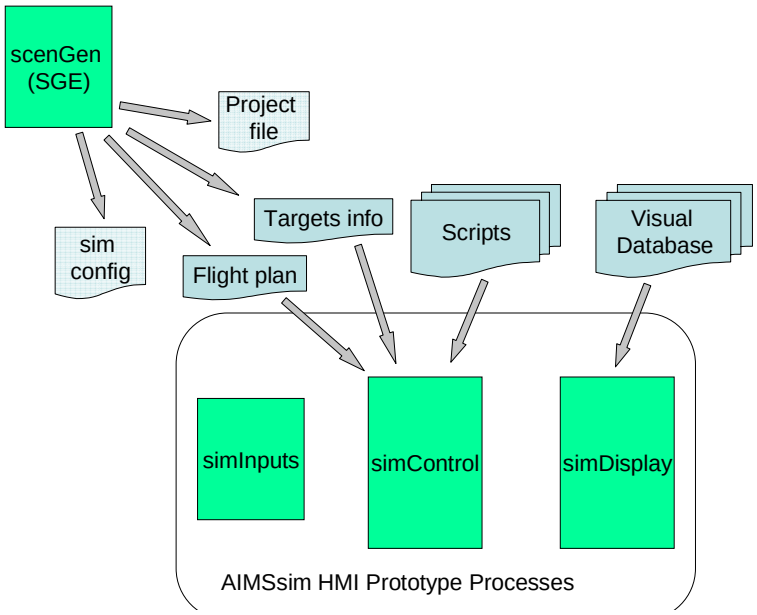


***Figure 9: System architecture when SGE used***

The SGE GUI allows the experimenter to set various simulation configuration parameters, such as the time-of-day, FLIR and AGTV noise, etc. However, *simControl* cannot currently read the simulation configuration file generated by the SGE (note how there is no arrow going from "sim config" box to *simControl*), so changing those parameters through the SGE will be a waste of time. The *only* UI parts of the SGE that are useful to AIMSsim experiments are the Targets pane and the Aircraft pane.

Due to the loose coupling between the SGE and the HMI, more details on the SGE subsystem are given only in the Annex B.

# 4 Limitations

## 4.1 General

The AIMS HMI Prototype has limitations that affect the overall fidelity of the AIMS simulation. Limitations are inherent to any computer simulation of a real world system and as such should be taken into consideration when employing the software as part of an empirical study.

The following sections provide an itemization of the limitations of the AIMS HMI Prototype. It should be noted that some limitations of the AIMSsim software are the result of the hardware and software associated with the computing platform (Microsoft Windows XP, OpenSceneGraph) on which the HMI Prototype is currently hosted. These hardware and software elements will be discussed in Section 5 - Operating Requirements.

## 4.2 Joystick input override

The FlyPanel joystick override capability has only been tested with Logitech USB joysticks. Old USB joysticks are unlikely to perform adequately.

Other USB joysticks should not require any software modification, but may require that the instructor redefine the mapping of input identifiers using the driver configuration software that comes with any modern USB joysticks (USB joysticks that are not configurable may not work on this system).

## 4.3 Sensor Simulation

The AIMSsim software includes the simulation of five types of sensors:

1. Active Gated TV (AGTV)
2. EO color camera, wide field of view (EO_W)
3. EO spotter, narrow field-of-view (EO_N)
4. Forward Looking Infrared (FLIR) imaging system
5. Thermal Infrared (THIR) imager

In fact, the implementation of these sensors would be best described as “emulation” rather than “simulation”. This is due to the fact that the sensor-like imagery being presented in the HMI Prototype is not the result of a physics based approximation of real EO and IR imaging apparatus. Rather the imagery is a “mock-up” of the sensor imagery, intended only to provide the “look and feel” of these EO and IR sensors for the purpose of providing a more complete HMI. As such, the HMI Prototype should

not be used to compare the relative effectiveness of EO and IR sensors, unless the shaders for each are modified to provide simulation rather than emulation.

Fog penetration is poorly modeled in AIMSsim. E.g. FLIRs have the ability to penetrate fog to some extent. The effect of having fog in the environment, if specified in the LUA scripts, does not affect the FLIR display. In this sense, the HMI FLIR has perfect penetration of fog. Better modeling of fog effects in FLIR displays may be worth considering for the future.

Similarly, image degradation due to noise and various physical phenomena is simulated very simply by adding a random component to the pixels in the EO and IR sensor displays. However, this does not take into account the effect of lower resolution of the real displays, pixilation, etc. Moreover, it uses the OpenGL shaders, which makes the noise depend on the texel (texture elements) size.

## 4.4 Measurement Units

All quantities in the system, such as distances, speeds, altitude, etc., are neither accurate nor precise representations of reality. E.g. the aircraft speed can be anything, even a speed faster than that of light. All physical quantities defined for an experiment are so defined for their role in supporting the objectives of the experiment, with no presumption of fidelity or physical realism.

## 4.5 Targets

The AIMS HMI Prototype includes a number of target models (vehicles, people, geometric primitives). These target models are limited insofar as they do not include any thermal information, forcing the IR sensors to use the same radiation information as the EO sensors, namely the visual spectrum. For this reason, the best IR effect is obtained for hot objects by representing their hot part with saturated colors (e.g. white).

An AIMSsim scenario is currently limited to a maximum of one hundred and fifty (150) targets and fifty (50) target types. Note however that targets can be created and destroyed during the experiment execution.

Another limitation is that targets that are clamped to the ground will be clamped to the highest Level-of-Detail (**LOD**) of the terrain model (see section 4.9.1 for a discussion of LOD). This may lead to some visual artefacts if the graphics display is using a lower-detail level of the terrain database: e.g. the target appearing slightly above or below the ground.

Another limitation is that simDisplay only gets updates from simControl at irregular intervals, because both processes are each running independently on the computer. If the display is running faster than the controller, then it is forced to “guess” where a moving target is, using dead reckoning. Once it gets an update from simControl, it corrects the target’s position. This could lead to jagged motion.

## 4.6 Aircraft Flight Simulation

The AIMS HMI Prototype simulates the movement of the sensor package through the simulated landscape as if it were attached to an imaginary search aircraft. The HMI Prototype does not, however, include a flight simulation model to control this movement. The HMI Prototype utilizes a simplistic path navigation algorithm that allows the movement of the simulated sensor "viewpoint" along straight-line segments between "waypoints" defined in the AIMSsim LUA scripts. The movement of the viewpoint is "smoothed" in the vicinity of the waypoints so as not to create a sudden change in direction as the path navigation algorithm transitions from one path segment to the next.

In real life, a search and rescue (SAR) aircraft would endeavour to maintain a consistent relative altitude (altitude above ground level or AGL) as it flies along a search path. This is not practical with the simplistic path navigation algorithm utilized by the HMI Prototype. As such, the HMI Prototype is limited insofar as it relies upon the absolute altitude (altitude above sea level or ASL) defined by the various waypoints to control the altitude of the simulated search aircraft.

## 4.7 Path Following

Path plans consist of a set of waypoints joined by straight lines. Targets and/or aircraft following the path plan will transition smoothly between the straight lines, whenever a "fillet radius" is defined for a waypoint. This causes the vehicles to miss the waypoints rather than pass "over" (or through) them. This behaviour, though not realistic from a pilot point of view, is used for two reasons:

1. It is easy for the experimenter to create paths since they can focus on the "corners" as a rough measure of where the vehicle should go, and the effect of a fillet radius is easy to envision while creating the path. The alternative requires the use of splines, which is far less intuitive;

2. This is how it was done in the old SGI-based system: the Performer™ library provided path following algorithms with the above behaviour; when the system was ported to Windows, the same behaviour was emulated in OpenSceneGraph.

Every vehicle (target or aircraft) travelling on a path will use the same attributes of the path plan, i.e. velocity, acceleration rate, etc. However, the scripting engine provides ways of copying paths and changing some of their properties.

## 4.8 Scripting

The scripting engine currently requires that scripts that load other scripts in the same folder explicitly specify the path to those scripts, relative to the folder from which *simControl* was executed. This is a common cause of bugs in the scripts when they are copied from an existing experiment into a new experiment folder: any changes to the (non-initialization) scripts in the new folder will not be seen if the experiment forgot to adjust the file paths in the experiment scripts, after copying the experiment. The two

functions to watch for are `dofile()` and `AddTransition()`, but a good technique is to search for all occurrences of the original experiment's folder name. E.g. if experiment A was copied to B, then all occurrences of "A" should be replaced with "B" in the scripts.

Setting attributes is currently rather tedious due to the low-level interfacing method used for the scripting engine. The scripting engine would benefit greatly from the use of one of the many open-source wrappers that automate the exposition of system variables, such as SWIG or ToLua++. E.g., instead of

```
Set("mapSensorHistoryState", "ENABLED")
ChangeTargetAttrib("target 1", "isTarget", "true")
```

the experimenter could simply write

```
map.sensorHistoryState = ENABLED
targets[1].is_target = true
```

The simDisplay scripting, having been added to the system more recently, makes use of the *tolua++* wrappers generator, such that the functions, modules, classes and variables available require less typing and allow for more checking.

## 4.9 Prototype Performance

With modern CPU's, motherboards and graphics cards, AIMSsim is capable of achieving a full 60 Hz runtime loop, using current scenarios. However, various aspects of terrains, targets and OpenGL shaders could cause this number to decrease under certain circumstances. If the system performance is degraded, one of the following effects might be to blame.

### 4.9.1 Effect of Terrain

The maximum achievable update rate is directly affected by the complexity of the visual terrain database that is employed for a particular scenario. The use of simplistic terrain databases will result in increased runtime performance, while the use of complex terrain databases will result in decreased runtime performance.

The complexity of a terrain database merits further discussion. A terrain database (as well as any 3D model) has two main characteristics that drive the hardware requirements for the computing platform on which it will be rendered: the number of polygons used to construct the model, and the amount of texture utilized by the model. Both of these should therefore be as small as possible, such that the experiment remains realistic.

Both the number of polygons and the amount of texture should be tailored to how close they are to the viewpoint (i.e. the sensor on the aircraft). OpenSceneGraph supports the concept of Level-of-Detail (LOD), which allows several versions of a terrain to be in the same terrain file. Each version has a different degree of detail, such that OpenSceneGraph will select which version of the terrain to show based on the distance between the terrain and the viewpoint. Adding LOD's to a terrain model can

improve performance dramatically. However, as described in an earlier section of this chapter, it can lead to some odd visual effects, such as a target clamped to the ground appearing to be above the ground.

### 4.9.2 Number of Target Objects

Similar to the performance limitations imposed by the complexity of the terrain database, the runtime performance of AIMSsim is affected by the number of target objects present in a scenario. Since each target object is composed of a limited number of polygons and each target object utilizes a portion of the system texture memory, the rendering demands on the workstation are increased with the addition of target objects.

Steps have been taken, however, to mitigate the performance degradation due to an increase in target objects. These steps include the use of Level-of-Detail (**LOD**) composition for target objects. LOD can be described as a strategy for reducing the number of polygons used to represent a geometric object by "switching" between numerous versions (of varying complexity) of an object at runtime, based on the distance of the viewpoint from the object. When the viewpoint is far away from an object, a representation of the object with a low number of polygons is utilized. As the viewpoint approaches the object, the low-polygon representation is replaced with a more complex version of the object (made up of more polygons). By trading off the complexity of the representation with the distance from the object, an attempt is made to balance the rendering capabilities of a system while presenting only the information that is required to accurately represent the appearance of a target object.

While this approach is helpful, it cannot address all possible scenarios. If for instance a large number of target objects (specifically those of a complex nature, i.e. a tank model) were assembled in a confined geographic area, LOD would not be helpful when the user "zoomed in" on this area. This is because the act of "zooming in" has moved the rendering viewpoint close to the collection of objects and forced them all to the highest level of detail.

### 4.9.3 Effect of Shaders

Shaders are machine code created at startup by the OpenGL library by compiling source code stored in shader text files into machine instruction. OpenGL is the low-level graphics API upon which OpenSceneGraph is built. The use of shaders is not absolutely necessary but adds tremendous freedom in modeling the various sensors. Without using shaders, the HMI display can be updated at over 60 Hz. When using the AGTV shader, this goes down to 20 Hz for an 800x600 display. The combined performance hit of having several shaders is not linear. However, shaders are a relatively new addition to OpenGL (ca. 2004), so graphics card vendors are still catching up to the various types of optimizations possible within the compiled shader code. The performance of shaders should improve dramatically over the next year, as graphics cards provide more hardware support for OpenGL Shading Language.

Every geometry object in the scene has a default GLSL Uniform named "nodeHasTexture" set to true. Objects that do not have texture will not be visible in the

AGTV display, unless their geometry defines this GLSL flag and sets it to false. Not all geometry modelling programs will allow you to define a GLSL Uniform as part of the geometry.

# 5 Operating Requirements

## 5.1 General

The following table lists the minimum system requirements to install and run the AIMSsim SGE and AIMSsim HMI Prototype.

***Table 1: Minimum Configuration Requirements***

| HARDWARE | SOFTWARE | MEMORY |
|---|---|---|
| ▪ CPU: DUAL Intel XEON 3.6GHz, 1MB Cache, 4GB RAM, PCI-Express,<br>▪ Graphics: BFG 6800Ultra Overclock Geforce 6800 256MB PCI-Express Dual DVI<br>▪ Monitor: Viewsonic VP201b Black 20.1" 1600x1200 HDTV LCD<br>▪ BG Systems FlyPanel | ▪ Microsoft Windows XP Pro, service pack 2<br>▪ OpenSceneGraph 0.9.9 or later, Expat 1.95.8, boost 1.32, LUA 5.0<br>▪ FLTK 1.1.7 (for the SGE)<br>▪ Visual Studio .Net 2003 (for modifications only) | ▪ 4GB memory<br>▪ 1GB disk space for geometry, binaries, and source code |

## 5.2 File and Folder Locations

Only the *simControl*, simDisplay and SGE have certain requirements:

- *simControl*: the *simInputs* and *simDisplay* executables, and all the DLLs, must be in the system PATH, or, if not, must be in the same folder as *simControl*.

- *simDisplay*: the visuals database is assumed to be located in the folder named by the AIMS_DATA environment variable, and is assumed to have the following structure:

```
AIMS_DATA_FOLDER
    target_map.txt
    shaders
    terrains
    targets
    textures
```

  All shaders are looked for in the *shaders* folder, all terrain geometry in the *terrains* folder, all target files specified in the *target_map.txt* file in the *targets* folder, and all textures in the *textures* folder.

- SGE: the scenarios can be stored anywhere, however the default location is as specified by AIMS_SCEN.

## 5.3 Environment Variables

Some run-time settings used by AIMSsim are independent of scenario and relate to "application configuration" rather than "scenario description". Those setting have been exposed through environment variables that should be set once in every user's environment, and need not be changed again. The following table summarizes the currently available environment variables for the AIMS HMI Prototype:

***Table 2: Environment Variables***

| ***Environment Variable*** | ***Description*** | ***Value*** |
|---|---|---|
| AIMS_DATA | Path to the AIMSsim database folder. This is a full path to the database described in section 5.2 -- File and Folder Locations. | String |
| AIMS_SCEN | Path to the AIMSsim scenarios folder, used by the SGE only. | String |
| AIMS_DPI | Space-separated pair of integers specifying dots-per-inch along x and y axes of the computer screen. Necessary for accurate rendering of moving map display. | int int |
| FB_EMULATOR | If set to something (anything), the *simInputs* will replace itself with a Java-based hardware emulator, useful when the FlyPanel is not available. Note that the emulator is only available to the staff at Greenley & Associates: it is an aid to development and is not "production quality", so it is not supported externally. | String |
| AIMS_ATD_CUES | If the variable exists (regardless of what it is set to), the Automatic Target Detection capabilities are turned on. If the variable does not exist, those capabilities are left off. | n/a |
| AIMS_ATD_CUES_IMG | The image file to use for type 2 cues. This file is assumed to be present in the %AIMS_DATA% /textures folder. | String |

# 6 Algorithms and Design

## 6.1 Components and Dependencies

Figure 10 represents the software components of AIMSsim and their dependencies. The green boxes represent programs that can be run, all others are libraries. The light blue boxes are in-house libraries, the dark blue boxes are Open Source libraries, typically based on the GPL license.

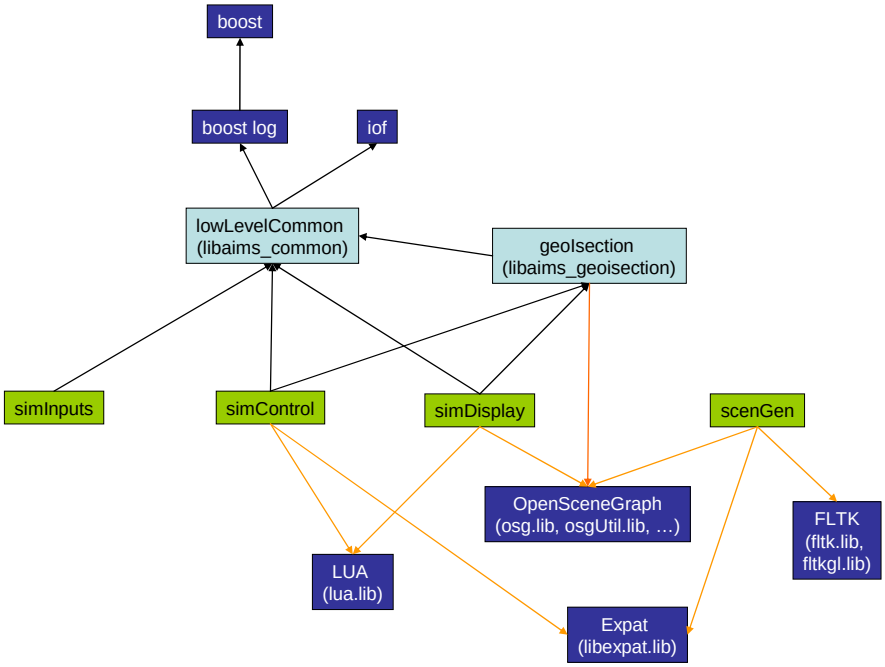


**Figure 10: AIMSsim components and their dependencies**

## 6.2 Simulation Loop

The simulation loop of *simControl* involves the following steps, which may be useful to know when defining timed events, periodic scripts, and other script tasks:

1. Get new events from *simDisplay*, put in event queue
2. Queue new events for run-timed events and flight-timed events (defined in scripts)
3. Copy shared memory written by *simDisplay* to local
4. Get new events from *simInputs* (digital inputs), put in event queue
5. Handle operator analog input
6. Update world entities (aircraft position and speed, targets, etc)
7. Execute periodic scripts

8. For each event in event queue: react to event (state transition, execute, etc.)
9. Commit the new state of the world to shared memory read by *simDisplay*
10. Send any messages queued to *simDisplay*, via pipe

Similarly, *simDisplay* uses a graphics update loop that does the following:

1. Copy shared memory written by *simControl* to local (new state of world)
2. React to any messages sent by *simControl* via pipe
3. Update the display
4. Commit any new computations to shared memory, in case *simControl* needs the results
5. Send any new events queued by *simDisplay*, to *simControl* via pipes

Step #3, "update the display", will involve more or less work depending on what is visible in the GUI: a dialog screen (e.g. the intro, start or exit screens), or the operational view. It basically consists of:

1. update the viewport dimensions (in case user has resized window)
2. handle keyboard and mouse events
3. for each subdisplay in display:
   a. update frame stamp
   b. update the scene graph
   c. cull
   d. draw

Note that a dialog panel only has one subdisplay, whereas the operational view has five (5).

## 6.3 Inter-process synchronization

*SimControl* and *simDisplay* are independent processes that communicate via pipes and shared memory. Each process uses atomic read/write operations on the whole shared memory segment at once, at the beginning or end of each simulation/graphics loop, so as to guarantee that the shared state is self consistent and doesn't contain a mixture of old and new data.

The separation of data exchange between the two processes between pipes and shared memory is somewhat unfortunate, since there is no synchronicity between pipes and shared memory. E.g., if simDisplay writes a message to a pipe, and immediately after it commits its new computations to shared memory, simControl may still see the shared memory update before it sees the pipe message. This has not been observed in practice but it is a possibility. The consequence is important only when the message being sent is related to the state being updated, as happens during an "isect request" to the simDisplay: simDisplay computes the isect, puts it in shared memory, and sends an event that the isection result is available. SimControl may get the "isection valid" event before its shared memory has been updated, which could cause the script that gets run when this event takes place to use garbage isection data.

## 6.4 Noise/Degradation

The degradation effect in the sensor displays uses a texture created at runtime from a fixed random seed. The texture coordinate of the active texture is used to look up the noise texture color, for every image fragment. The noise is greyscale and stored in the shader file *noise2DRed.frag*. It requires only the noise amplitude and a random offset so that the noise appears to change in time on the display. See the *applyNoise*() functions defined in the shader file for more details. A better technique would be to use in-memory rendering of the scenery image, and apply a noise function to the image, before displaying it.

## 6.5 Fog

The prototype can provide a distance based fog to restrict the viewing distance of the sensor images. This fog is a per-pixel linear OpenGL Fog. However it is implemented as an OpenGL shader, hence it could be improved easily without requiring any software modifications. The parameters for the fog can be set using the *fogDistance*, *fogOnset*, and *fogColor* Lua variables. The first variable sets the maximum viewable distance, in metres, and the second sets the near edge of fog. If *fogDistance* is not set, no fog is provided. The fog color, greyscale, is the product of the *timeOfDay* value and *fogColor* value.

## 6.6 LOD (TFLOD)

The simulation provides a few methods to tune level of detail used for the terrain. First, the *agtvLODScale* and *flirLODScale* LUA variables allow the LOD scale for the simulations to be set individually. A value of 1 means the LOD will behave as specified by the terrain. A value less than 1 will increase the switching distance, a value of more than 1 will decrease the switching distance. A value of 0 means that the highest level of detail will always be used. See the OpenGL Performer documentation for a more complete description of this feature.

On computers with a graphics card that supports Terrain Fade Level of Detail (TFLOD), this option will be enabled by default. This option smoothes the transition between levels of detail.

## 6.7 AGTV Illuminator

The AGTV illuminator is modelled differently if fog is present. Without fog, the illuminator is assumed to be on with auto-gain control such that the area of the display outside the illuminated area appears dimmed, while the scene in the illuminated area is untouched.

When fog is toggled on, the illuminator is modelled as “seeing through” the fog. The illuminated area appears without fog, the non-illuminated area appears with normal, undimmed fog.

Objects that are retro-reflective will become saturated white when they enter the illuminated area.

## 6.8 Auto-tracking

The auto-tracking feature is implemented via LUA scripting. It currently makes use of AIMSsim exported functions to reset periodically (several times per second) the camera pod heading and pitch, so as to be "pointing" towards the ground position that was in the Line of Sight at the time auto-tracking was started. It is therefore only an approximation:

- The center point of the AGTV display must be on some object (the sky won't work)
- If the target that is being auto-tracked becomes hidden behind a mountain the tracking will not be lost
- The "tracker" works independently from the AGTV or FLIR displays and does not depend on noise level etc

As well, setting the heading and pitch of the camera pod can cause a small amount of jitter, since small numerical errors in the angles will result in large variations of the line of sight intersection on the terrain. AIMSsim provides a mode to directly set the "point to look at" instead of setting the heading and pitch.

## 6.9 Terrain clamping and following for targets

All targets get automatically clamped to terrain underneath or above them when added to the world via the Lua initialization script. Clamping refers to changing their Z coordinate so as to position them on the ground. This allows an experimenter to not worry about target height and makes a scenario adaptable to different terrain databases. Targets that are moved in the XY plane (via scripting) are automatically re-clampled to the ground at their new latitude-longitude position. This means that targets can never be floating in mid-air.

In the most recent developments, *simControl* was given the ability to query a geometry database to find out where the targets should be clamped. This allows scripts to know immediately (without having to wait for *simDisplay* to do the computation) where a target is going to be clamped.

Note that the current implementation uses OpenSceneGraph to implement the database reader, and to compute terrain height at a given position. Technically, simControl is completely unaware of this fact: from its point of view, it is using a library that has the ability to read a geometry file and return height at a given XY. This library, called *geoIsection*, uses the highest level of detail available from the terrain database, because this is how OpenSceneGraph operates. *SimControl* doesn't care and doesn't know.

*SimDisplay* currently uses the same library, *geoIsection*, to clamp the targets to the ground, independently of *simControl*. It does this because it uses dead reckoning to estimate where the target probably is in the XY plane, if an update from *simControl* is not yet available. Unfortunately, OpenSceneGraph uses the highest level of detail available from the terrain database when doing the geometry calculations. This may be a different level of detail than seen in the operational screen by the human operator, since the level of detail in use visually is controlled by the underlying graphics library scene viewer, based on the distance of the viewpoint from the observed geometry. There is hence the potential for the target to disappear under the ground at the lower level of detail. This will not, in general, be a problem, but should be kept in mind if some odd behaviour is observed.

Another consequence of separation of model (*simControl*) and view (*simDisplay*) is that operations based on "designating" a target (such operations take place in the scripts in simControl) could fail if the target is visible at a different height due to the dead reckoning, level of detail used in the visuals, etc. This is an inherent limitation of any simulation system with this kind of architecture.

Finally, note that targets moved along a non-flat ground cannot maintain a constant speed, because the speed affects how "far" along the surface the target can move, yet the slope of the surface changes at every point of the "journey".

## 6.10 Lua Scripting

The Lua scripting capabilities of the system are implemented differently in *simControl* and in *simDisplay*. In *simControl* the legacy approach is used (essentially identical to the technique used in the old SGI-based system): explicitly use the Lua C API to register exported functions. In *simDisplay*, the *tolua++* tool is used to automatically generate all the C API calls to make specified application functions, variables, constants and classes available to the scripting engine. The *tolua++* website is at http://www.codenix.com/~tolua/. Version 1.0.92 of *tolua++* was used for the current version of the AIMSsim system.

The latter technique is far superior in that it is far simpler, much less error prone, yet allows more comprehensive access to the application's data and types, thereby obviating the need for cumbersome quotation marks for constants. Implementing this functionality with the straight C API of Lua would be tedious and error prone and would require much work to reach the error checking level that *tolua++* allows for the system.

## 6.11 ATD Cues

The system supports assigning a different cue to each target. The box and image cues can be assigned to any target, whereas the color-inversion cue can only be assigned to

up to three targets. The visual characteristics (e.g. frame color in the case of box cue) of cues cannot be changed. Cue positions are updated at every time frame, always using the target position as the center point of their own geometry. Cues are only displayed in the AGTV sensor display.

Cues are visible only when their associated target is visible in the sensor display. At every cue update cycle, a line of sight test is performed from the aircraft position to the cue's target in the world. If geometry collision is detected, it is understood that the target is currently behind geometry and hence not visible.

Note that cues are only created at startup. Hence adding new targets after initialization will not create new cues. All targets that require a cue must be created at initialization time but can be made invisible until such time as the target is needed in the experiment.

# References

1. Baker, K., & Youngston, G. (2006). *Advanced Integrated Multi-sensor Surveillance (AIMS) Operator-machine interface definition study*. CAE Professional Services. Defence Research and Development Canada - Toronto Contract Report: DRDC Toronto CR 2006-242.

2. McFadden, S., Crebolder, J., & Larochelle, V. (2006). *Development of an operator-machine interface for ELVISS Final Report*. Defence Research and Development Canada - Toronto Technical Report: DRDC Toronto TR 2006-055.

3 Schoenborn, O. (2007a). *AIMSsim User Manual*. CAE Professional Services, Ottawa, Ontario. Defence Research and Development Canada - Atlantic: DRDC Atlantic CR 2007-300.

4. Schoenborn, O. (2007b). *Human Factors Research Task 2006-111: AIMSsim Feature Development II*. CAE Professional Services, Ottawa, Ontario. Defence Research and Development Canada - Atlantic: DRDC Atlantic CR 2007-302.

# Annex A: Scenario Generation Environment (SGE)

The SGE is an ideal environment in which to create flight paths and position targets as it gives a visual representation of a specified terrain, targets, and a flight plan, and has several types of flight plans predefined and parameterized.

The current version of the SGE creates 4 data files in a XML format. Only two of these files can be read directly by the AIMSsim application: the flight plan file, whose parameters are edited on the "Aircraft" panel, and the targets definitions file, whose parameters are edited on the "Targets" panel. Both panels are described in the following sections, B.1 and B.2. *All other panels* contain parameters that cannot be read by AIMSsim, so those panels are of little use for experiment development. They are described briefly in this Annex's sections B.3 and B.4 only for historical reasons, and should not be used as a reference.

The SGE comprises a graphical user interface (GUI) featuring a moving-map display of the geographic scenario area, as well as graphical controls providing the means to manipulate scenario parameters. The SGE GUI is depicted in Figure 11.

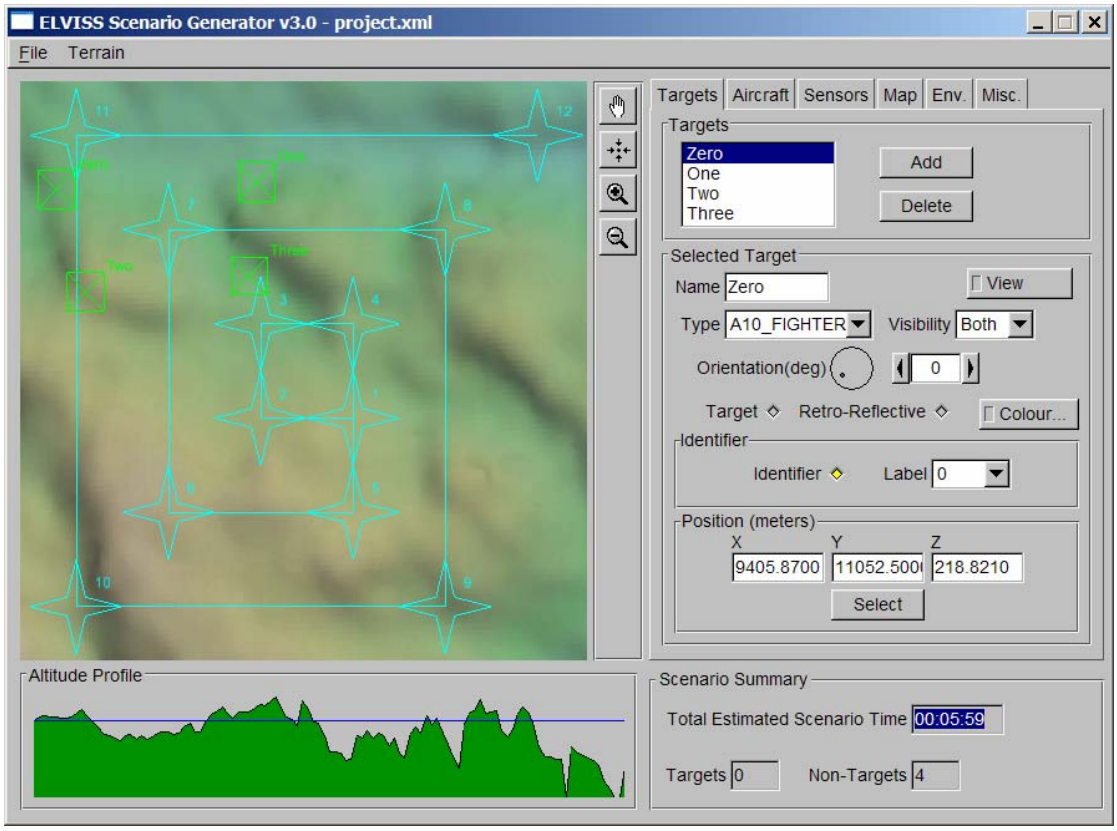


***Figure 11. SGE User Interface***

The SGE allows the experimenter to control the following scenario elements:

a. Position and appearance of target objects on the scenario landscape;

b. Flight plan of the simulated search aircraft;

c. Configuration of the sensor package and HMI screen layout; and

d. Environmental parameters controlling the effectiveness of the simulated sensors.

In addition, the SGE provides the following features:

a. Real-time manipulation of the moving-map display (zoom and pan);

b. Graphical depiction of the aircraft altitude profile throughout the scenario;

c. Estimation of scenario duration; and

d. Total number of targets and non-targets defined in the scenario.

## A.1 Manipulation of Targets

The SGE provides an interface whereby an experimenter may manipulate target objects that will form part of an experimental scenario. At the highest level, the SGE allows a user to "Add" and "Delete" targets from the scenario. Once a target has been added to the scenario, a user may modify various parameters that affect the position and appearance of the target object when viewed in the HMI prototype. Table 3 describes the modifiable target parameters and defines the valid range of values for each parameter as well as the units of each parameter, where applicable.

***Table 3. SGE Target Parameters***

| PARAMETER | VALID RANGE | UNITS |
|---|---|---|
| Name | Any alpha-numeric string (maximum of TBD characters) | N/A |
| Target | Boolean value | N/A |
| Retro_reflective | Boolean Value | N/A |
| Type | Pick list (see Appendix A of the User Manual for a list) | N/A |
| Visibility | Pick list (Both, AGTV Only, FLIR Only) | N/A |
| Orientation | 0 to 359 | Degrees |
| Identifier | Boolean value | N/A |
| Identifier Label | Pick list (A to F, 0 to 9) | N/A |

| | | |
|---|---|---|
| Position (x, y, z) | Single precision floating point value | Meters |
| AGTV Colour Adjust | Boolean value | N/A |
| AGTV Colour | Single precision floating point value | N/A |
| FLIR Colour Adjust | Boolean value | N/A |
| FLIR Colour | Single precision floating point value | N/A |

A user may modify the position of a selected target by two methods: entering the discrete values for “x”, “y” and “z” or by selecting a location on the moving-map display, after clicking the “Select” button. When entering discrete values, the user must press ‘Enter’ (after the entry of each value) for the parameter changes to be registered to the target in question.

## A.2 Manipulation of Flight Plan

The SGE allows the experimenter to modify the sequence of waypoints (or flight plan) of a simulated search aircraft to which the simulated sensor package is mounted. The flight plan interface allows the experimenter to select between three different methods of creating a flight plan. They are:

a. User defined flight plan. The user may interactively “Add” and “Delete” waypoints from the flight plan. For each waypoint the user may modify the position (x, y, altitude) and the speed of the aircraft at that waypoint.

b. Creeping line search pattern. The user may create a flight plan based on an algorithm that builds a “Creeping Line Ahead” flight pattern. The user selects a “Start Point” for the pattern and provides additional parameters necessary for the SGE to automatically construct the flight plan.

c. Expanding square search pattern. The user may create a flight plan based on an algorithm that builds an “Expanding Square” flight pattern. The user selects a “Start Point” for the pattern and provides additional parameters necessary for the SGE to automatically construct the flight plan

Tables 4, 5 and Table 6 describe the modifiable flight plan parameters for each of the flight plan creation methods mentioned above.

***Table 4. SGE User Defined Flight Plan Parameters****

| PARAMETER | VALID RANGE | UNITS |
|---|---|---|
| Waypoint Position (x, y, altitude) | Single precision floating point value | Meters |
| Speed | 0 to 200 | Knots |

*These parameters apply to each waypoint in the flight plan.

***Table 5. SGE Creeping Line Flight Plan Parameters***

| PARAMETER | VALID RANGE | UNITS |
|---|---|---|
| Start Position (x, y, altitude) | Single precision floating point value | Meters |
| Speed | 0 to 200 | Knots |
| S | 0.1 to 10.0 | Kilometres |
| Leg Length | 0.1 to 20.0 | Kilometres |
| # of Legs | 1 to 150 | N/A |

***Table 6. SGE Expanding Square Flight Plan Parameters***

| PARAMETER | VALID RANGE | UNITS |
|---|---|---|
| Start Position (x, y, altitude) | Single precision floating point value | Meters |
| Speed | 0 to 200 | Knots |
| S | 0.1 to 10.0 | Kilometres |
| # of Legs | 1 to 150 | N/A |
| Direction | Pick list (CW, CCW) | N/A |

## A.3 Manipulation of Sensors

The SGE provides the user with the capability to modify the characteristics of the simulated AIMS sensors. These characteristics affect the window layout configuration to be used by the HMI prototype, as well as the Field Of View (FOV) or zoom of the individual sensors. Table 7 describes the modifiable sensor parameters and defines the valid range values and units for each parameter, as applicable.

***Table 7. SGE Sensor Parameters***

| PARAMETER | VALID RANGE | UNITS |
|---|---|---|
| Sensor Window Configuration | Pick list (Both Equal, AGTV Primary, FLIR Primary, AGTV Only, FLIR Only) | N/A |
| Slave Sensor FOV | Boolean value | N/A |
| AGTV Zoom (min/max) | 0.5 to 40.0 | Degrees |
| AGTV Zoom Method | (Continuous/ Discrete) | N/A |
| FLIR Zoom (min/max) | 0.5 to 40.0 | Degrees |
| FLIR Zoom Method | (Continuous/ Discrete) | N/A |
| Simulate CCD | Boolean value | N/A |
| Default AGTV Beam Width | Pick list (Wide, Narrow) | N/A |
| AGTV Narrow Beam Size | Pick list (2, 5) | Degrees |
| AGTV Wide Beam Size | Pick list (10, 15, 20, 25, 30, 35) | Degrees |

## A.4 Manipulation of Additional Scenario Parameters

The SGE allows you to control additional scenario settings. These miscellaneous settings control the default configuration of the HMI Prototype display and of the HMI control hardware. Table 8, Table 9, and Table 10 describe the modifiable miscellaneous scenario parameters and define the valid range values and units for each parameter, as applicable.

***Table 8. SGE Map Parameters***

| PARAMETER | VALID RANGE | UNITS |
|---|---|---|
| Map Mode | Pick list (No Map, 2D Paper, 2D Terrain, 2D Shaded, Immersed, Immersed Shaded, Tethered, Tethered Shaded) | N/A |
| Map Scale | Range (1:1000, 1:250000) | N/A |
| Map Orientation | Pick list (North Up, Aircraft Up, Camera Up) | N/A |
| Aircraft Symbol | Pick list (Rotary Wing, Fixed Wing, Pointer, No Icon) | N/A |

| | | |
|---|---|---|
| Map Type | Pick List | N/A |
| Colour | Pick List(Colour, Greyscale) | N/A |
| Map Slave FOV to Sensor | Boolean | N/A |
| Map FOV | 1 to 120 | Degrees |

***Table 9. SGE Environment Parameters***

| PARAMETER | VALID RANGE | UNITS |
|---|---|---|
| Time of Day (TOD) | 0.00 to 1.00 | N/A |
| Degradation Visibility | Pick List (Clear Degraded) | N/A |
| Degradation AGTV effect | 0.00 to 1.00 | N/A |
| Degradation FLIR effect | 0.00 to 1.00 | N/A |

***Table 10. SGE Miscellaneous Parameters***

| PARAMETER | VALID RANGE | UNITS |
|---|---|---|
| Scan Enabled | Boolean value | N/A |
| Scan Rate | Pick list (1.5, 3.0, 6.0) | Deg/s |
| Scan Width | Pick list (30, 60, 90) | Deg |
| Joystick Mode | Pick list (Aircraft, Cursor) | N/A |
| Zoom Control Mode | Pick list (Forward=Zoom In, Forward=Zoom Out) | N/A |
| Memory Recall Mode | Boolean | N/A |

## A.5 Target Preview

The SGE provides the user with the capability to view the placement and orientation of an individual target on the scenario landscape. This is now a mode of operation from within the main window of the SGE application. This mode can be toggled by pressing/releasing the “view” button after selecting a target. Once invoked, the selected target is displayed in relation to the terrain where it was placed. By using the left and right mouse buttons, the target (as well as the corresponding scene) can be manipulated in 3D space. In addition, the target’s orientation can be manipulated using the orientation dial control in the SGE window. The target view mode is depicted in Figure 12.

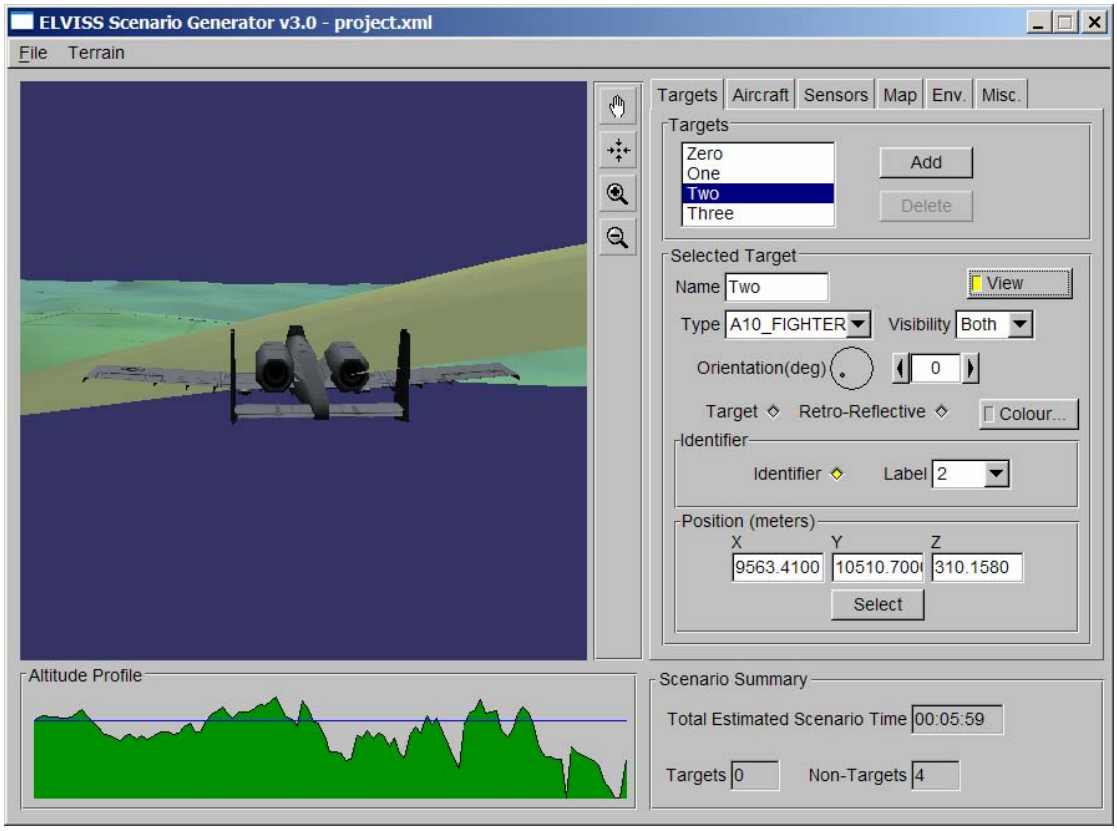


***Figure 12. SGE Target Preview Application (Manual Mode)***

## A.6 File Input/Output

The SGE provides the user with the capability to "load" and "save" entire scenarios as well as individual scenario elements. This merits some discussion. A "scenario", as defined in the context of this software, is comprised of several scenario elements. Scenario elements include a target list, a flight plan and a prototype configuration.

A target list comprises a sequence of target definitions containing the parameters defined in Table 3. A flight plan consists of a sequence of waypoints representing the parameters defined in Tables 4, 5 and Table 6. A prototype configuration consists of configuration parameters defined in Table 7, 8, 9 and 10 as well as a reference to a terrain landscape. By defining new scenario elements or by mixing and matching existing scenario elements, the user can create an infinite number of scenario possibilities.

The input/output file format specifications for the various file types supported by the SGE are provided in Section 6 of the *AIMS Software Prototype User Manual*.

# Bibliography

1. Gamble, M. (2001). *ELVISS Software Prototype - System Manual.* The HFE Group. Defence Research and Development Canada - Toronto Contract Report: DRDC Toronto CR 2001-029.

2. Gamble, Murray G. (November 10, 2000). *Proposal to Modify ELVISS Prototype Software. Volume One - Technical and Management Proposal*. The HFE Group.

3. Neal, B. (April 28, 1999). *ELVISS Human Engineering Design Approach Document – Operator*. Canadian Marconi Company, Human Factors Engineering Aerospace Division. Defence Civil Institute of Environmental Medicine Contract Report Number: DCIEM-CR-1999-078CMC-1000-1182.

4. Schoenborn, O. (2006a). *AIMSsim version 2.2.1 User Manual*. CAE Professional Services, Ottawa, Ontario. Defence Research and Development Canada - Atlantic: DRDC Atlantic CR 2006-280.

5. Schoenborn, O. (2006b). *AIMSsim version 2.2.1 System Manual*. CAE Professional Services, Ottawa, Ontario. Defence Research and Development Canada - Atlantic: DRDC Atlantic CR 2006-270.

6. Schoenborn, O. (2006c). *Herc SAR Task 106: AIMS Feature Development Final Report*. CAE Professional Services, Ottawa, Ontario. Defence Research and Development Canada - Atlantic: DRDC Atlantic CR 2006-278.

This page intentionally left blank

# List of abbreviations and acronyms

| | |
|---|---|
| AGL | Above Ground Level (refers to altitude) |
| AGTV | Active Gated TV |
| AIMS | Advanced Integrated Multi-sensor Surveillance |
| AIMSsim | AIMS simulator |
| ASCII | American Standard Code for Information Interchange |
| ASL | Above Sea Level (refers to altitude) |
| ATD | Automatic Target Detection |
| DND | Department of National Defence |
| DRDC | Defence Research and Development Canada |
| ELVISS | Enhanced Low-Light Level Visible and Infrared Surveillance System |
| EO | Electro-Optic |
| FLIR | Forward Looking Infrared |
| FLTK | Fast Light Tool Kit |
| FOV | Field of View |
| FSM | Finite State Machine |
| GLSL | OpenGL Shading Language |
| GPL | General Public License |
| GUI | Graphical User Interface |
| HMI | Human Machine Interface |
| IR | Infrared |
| LOD | Level of Detail |
| LOS | Line of Sight |
| MMD | Moving Map Display |
| N/A | Not Applicable |
| PC | Personal Computer |
| SAF | Semi-Automated Forces |
| SAR | Search and Rescue |
| SGE | Scenario Generation Environment |
| SGI | Silicon Graphics, Inc. |
| TOD | Time of Day |
| UI | User Interface |
| VAPS | Virtual Applications Prototyping System |
| VPI | Virtual Prototypes, Inc. |
| Wx | Weather |

This page intentionally left blank

# Distribution list

Document No.: DRDC Atlantic CR 2007-301

LIST PART 1: Internal Distribution by Centre:

5 DRDC Atlantic Library
2 DRDC Atlantic Scientific Authority

7 TOTAL LIST PART 1

LIST PART 2: External Distribution by DRDKIM

1 Jocelyn Keillor
DRDC Toronto
PO Box 2000
Toronto, Ontario M3M 3B9
Canada
1 DRDKIM

2 TOTAL LIST PART 2

**9 TOTAL COPIES REQUIRED**

This page intentionally left blank

**DOCUMENT CONTROL DATA**
(Security classification of the title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (The name and address of the organization preparing the document, Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's document, or tasking agency, are entered in section 8.)

Publishing: DRDC Atlantic

Performing: CAE Professional Services, 1135 Innovation Dr., Ottawa, ON, K2K 3G7

Monitoring:

Contracting:

2. SECURITY CLASSIFICATION (Overall security classification of the document including special warning terms if applicable.)

UNCLASSIFIED

3. TITLE (The complete document title as indicated on the title page. Its classification is indicated by the appropriate abbreviation (S, C, R, or U) in parenthesis at the end of the title)

AIMSsim version 2.3.4 − System Manual (U)

4. AUTHORS (First name, middle initial and last name. If military, show rank, e.g. Maj. John E. Doe.)

O. Schoenbom; P. Lachance; N. Bahramifarid

5. DATE OF PUBLICATION (Month and year of publication of document.)

January 2008

6a NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.)

56

6b. NO. OF REFS (Total cited in document.)

4

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical document, technical note or memorandum. If appropriate, enter the type of document, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Contract Report System manual for a multi−sensor simulator research platform that includes sensor interface and controls and airborne scene. Associated documents: DRDC Atlantic CR 2007-300 − User Manual, DRDC Atlantic CR 2007-302 − AIMSsim Final Report.

8. SPONSORING ACTIVITY (The names of the department project office or laboratory sponsoring the research and development − include address.)

Sponsoring:

Tasking: DRDC Atlantic
DRDC Atlantic

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant under which the document was written. Please specify whether project or grant.)

13dx

9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)

W7711−047904/TOR/001

10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document)

DRDC Atlantic CR 2007−301

10b. OTHER DOCUMENT NO(s). (Any other numbers under which may be assigned this document either by the originator or by the sponsor.)

11. DOCUMENT AVAILABILITY (Any limitations on the dissemination of the document, other than those imposed by security classification.)

Unlimited distribution

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11), However, when further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.))

Unlimited announcement

**DOCUMENT CONTROL DATA**
(Security classification of the title, body of abstract and indexing annotation must be entered when the overall document is classified)

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

(U) This system manual provides an overview of software developed to support the empirical investigation of a simulated user interface for an Advanced Integrated Multi−sensor Surveillance (AIMS) system (formerly known as the Enhanced Low−Light Level Visible and Infrared Surveillance System – ELVISS). The AIMS system is an electro−optical imaging system being developed by Defence Research and Development Canada (DRDC) – Valcartier to enhance the capability of search and rescue (SAR) crews to operate effectively at night and in degraded weather conditions. In order to ensure that a SAR operator would be able to use the system effectively and with a minimal amount of training, a prototype human−machine interface (HMI) was developed to permit evaluating design concepts. The latest development phase added sensor controller options and a configurable display interface for evaluating interface design concepts.

(U) Le présent manuel de système donne un aperçu du logiciel élaboré pour prendre en charge l'examen empirique d'une interface utilisateur simulée d'un système perfectionné de surveillance multi−capteurs intégré (AIMS) (anciennement appelé Système perfectionné de surveillance à intensification de lumière visible et à infrarouge ou ELVISS). Le système AIMS est un système d'imagerie électro−optique en cours de développement à Recherche et développement pour la défense Canada (RDDC) – Valcartier pour améliorer la capacité des équipages de recherche et de sauvetage (SAR) à travailler avec efficacité la nuit et par intempérie. Afin de garantir qu'un opérateur du service SAR sera en mesure d'utiliser le système efficacement et moyennant une formation minimale, un prototype d'interface humain−machine (IHM) a été développé pour évaluer les principes de conception. La dernière phase de développement consistait à ajouter des options de contrôleur des capteurs et une interface d'affichage configurable pour évaluer les principes de conception d'interface.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

(U) simulation; airborne; surveillance; search and rescue; electro−optical sensor; operator−machine interface

This page intentionally left blank.

**Defence R&D Canada**

Canada's leader in defence
and National Security
Science and Technology

**R & D pour la défense Canada**

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale


**www.drdc-rddc.gc.ca**